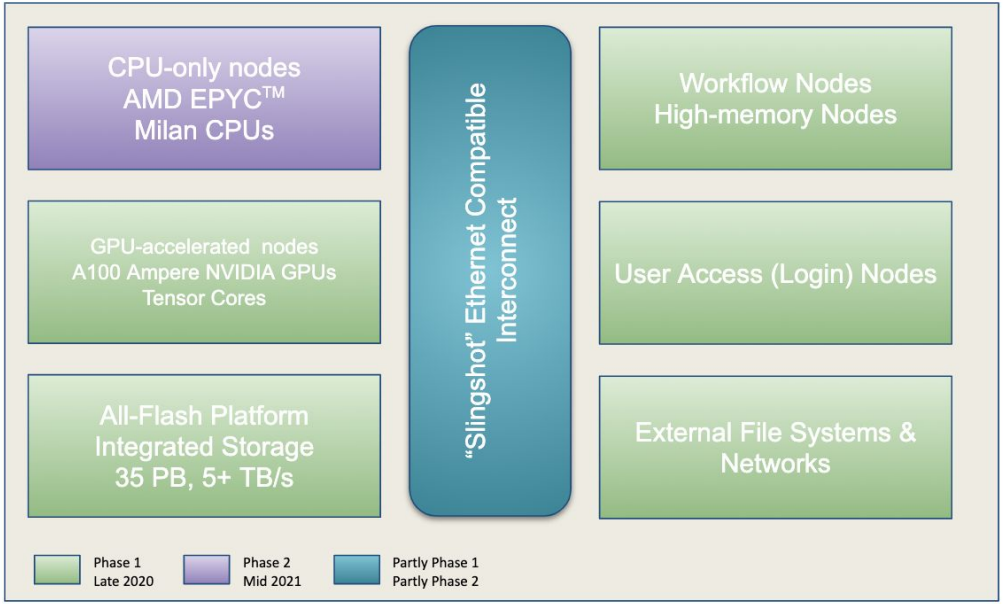
Preparing Apps for the Perlmutter System and beyond at NERSC

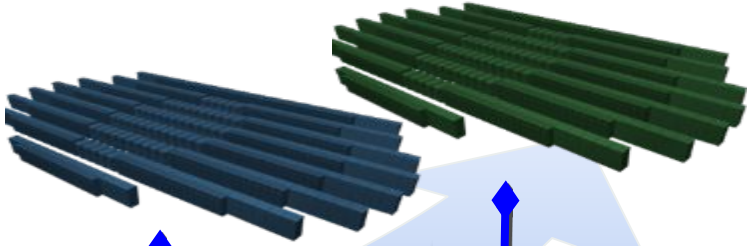
Jack Deslippe
NERSC
September, 2020

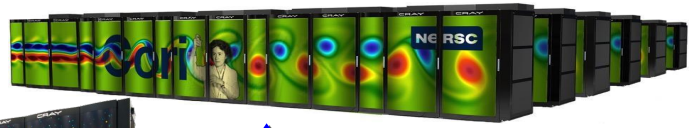
Perlmutter: a System Optimized for Science

- NVIDIA A100-accelerated and CPU-only nodes meet the needs of large scale simulation and data analysis from experimental facilities
- Cray “Slingshot” - High-performance, scalable, low-latency Ethernet-compatible network
- Single-tier All-Flash Lustre based HPC file system, 6x Cori’s bandwidth
- Dedicated login and high memory nodes to support complex workflows



NERSC System Roadmap





NERSC-7: Edison Multicore CPU

2013

NERSC-8: Cori Manycore CPU
NESAP Launched: transition applications to advanced architectures

2016

NERSC-9: CPU and GPU nodes
Continued transition of applications and support for complex workflows

2020

NERSC-10: Exa system

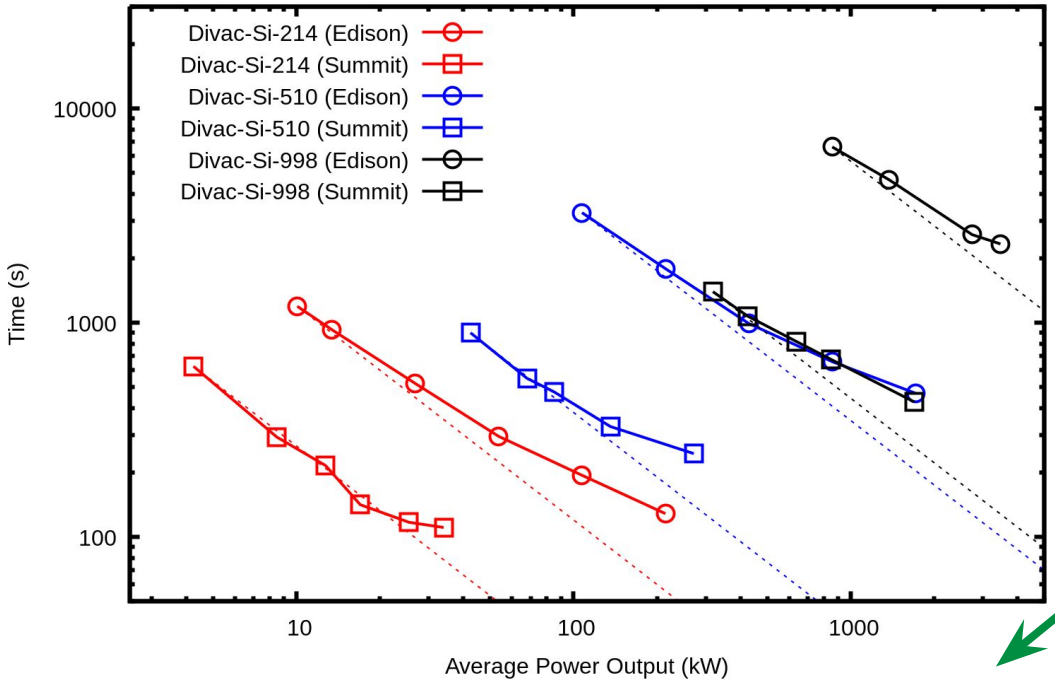
2024

NERSC-11: Beyond Moore

2028

Increasing need for energy-efficient architectures

Why GPUs



Improving Energy Efficiency

CPU Nodes

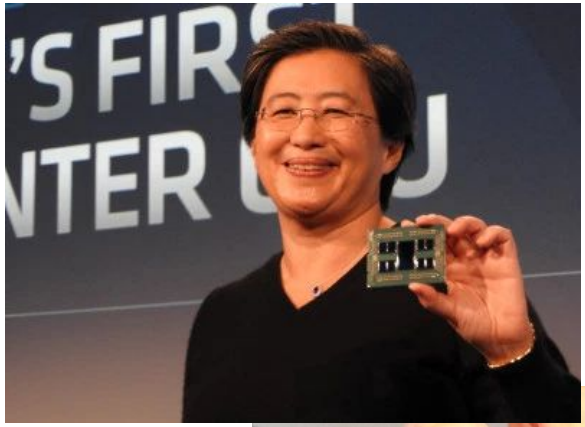
AMD “Milan” CPU

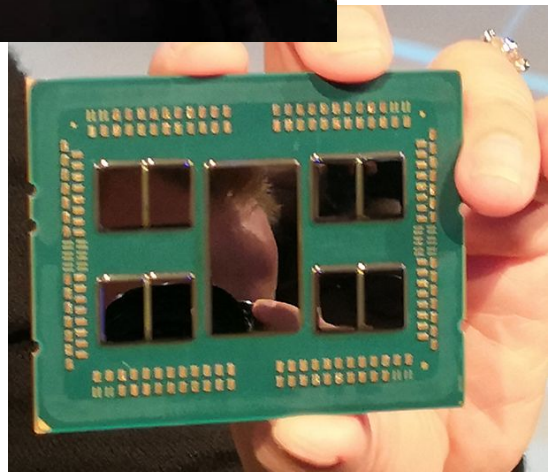
- ~64 cores
- “ZEN 3” cores - 7nm+
- AVX2 SIMD (256 bit)

>=Rome specs

8 channels DDR memory

~ 1x Cori





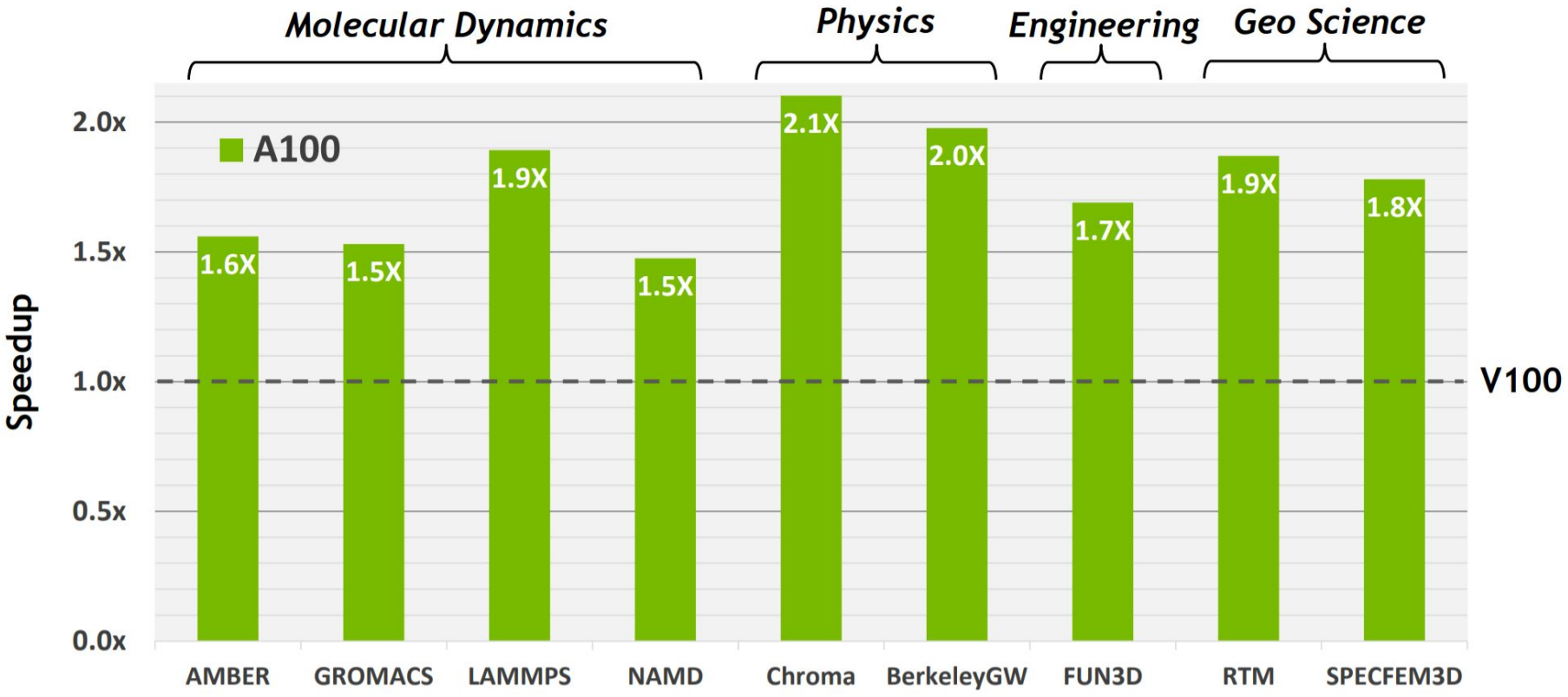
GPU Nodes

- 4x NVIDIA Ampere (A100) GPUs
- 1 AMD Milan CPU

- NVLINK-3 (Between 4 GPUs)
- FP16, TF32, FP64 Tensor Cores
- GPU direct
- Multi-Instance GPU (MIG)

	V100	A100
FP64 Peak	7.5 TF FMA	19.5 TF TC (9.7 TF FMA)
FP16 Peak	125 TF TC	312 TF TC
SMs	80	108
Memory BW	900 GB/s	1555 GB/s
Memory Size	16 GB	40 GB
L2 Cache	6 MB	40 MB
Shared Mem. / SM	96 KB	164 KB

A100 vs V100


Molecular Dynamics
Physics
Engineering
Geo Science
Speedup
2.0x
1.5x
1.0x
0.5x
0.0x
A100
V100
1.6X
1.5X
1.9X
1.5X
2.1X
2.0X
1.7X
1.9X
1.8X
AMBER
GROMACS
LAMMPS
NAMD
Chroma
BerkeleyGW
FUN3D
RTM
SPECFEM3D

Application Readiness Strategy for Perlmutter

NERSC's Challenge

How to enable NERSC's diverse community of 7,500 users, 750 projects, and 700 codes to run on advanced architectures like Perlmutter and beyond?

Programming Models on Perlmutter

We will support and engage our user community where their apps are (PGI, GCC, LLVM):

CUDA: MILC, Chroma, HACC ...

CUDA FORTRAN: Quantum ESPRESSO

OpenACC: VASP, E3SM, MPAS, GTC, XGC ...

Kokkos: LAMMPS, PELE, Chroma ...

Raja: SW4

+ (OpenMP, Planned Support for HIP, DPC++)

Application Readiness Strategy for Perlmutter

Users Consistently Tell Us They Care a Lot About

1. Portability - They (and their users) have accounts at HPC Centers all over the world.

2. Longevity - They don't want to have to rewrite their code for every system we or others deploy.

OpenMP for GPUs

- NERSC is collaborating with NVIDIA to enable OpenMP GPU acceleration in PGI (NVIDIA SDK) compilers
  - Co-design with application requirements
- OpenMP 5.0+ improvements for accelerators
  - Unified memory support
  - Implicit declare target
- Tell us your experience!
  - Techniques that work? Failures?

Our Messaging

1. **Understand your code performance. It's easier than you think and what you learn will help you everywhere.**

Roofline for Performance Analysis

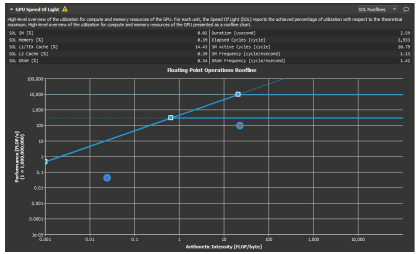
Users Want to Know:

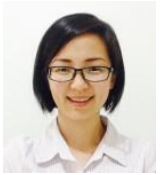
- What part of my code should I move to GPU?
- How do you know what HW features to target: HBM, Latency Hiding, Shared Mem, Packed Warps...
- How do you know how your code performs in an absolute sense and when to stop?

Progress Towards Roofline on GPUs:

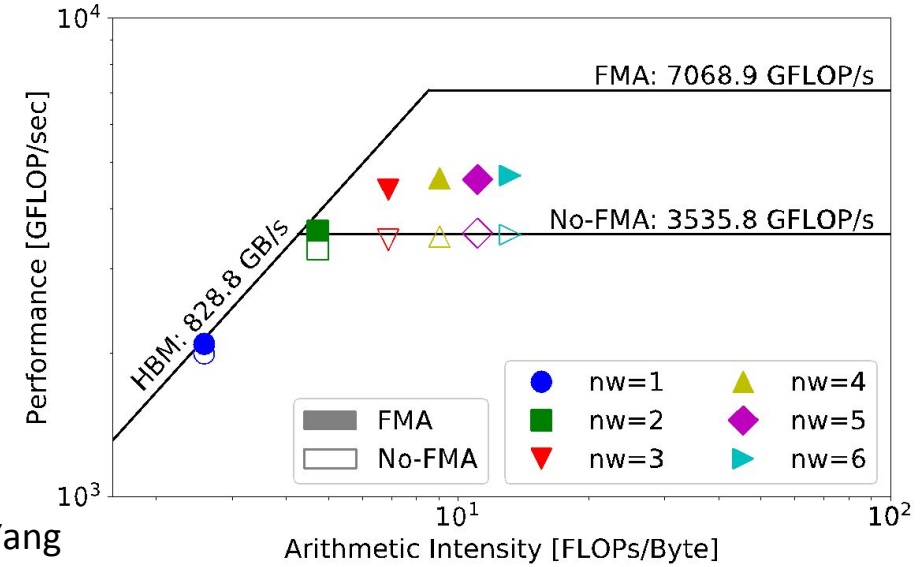
Worked with NVIDIA to ensure NVProf/NSight can collect all required metrics including data motion from multiple levels:
L1/Shared, L2, DRAM, Host DRAM *etc*.

Roofline is now Integrated in NVIDIAs NSIGHT tool



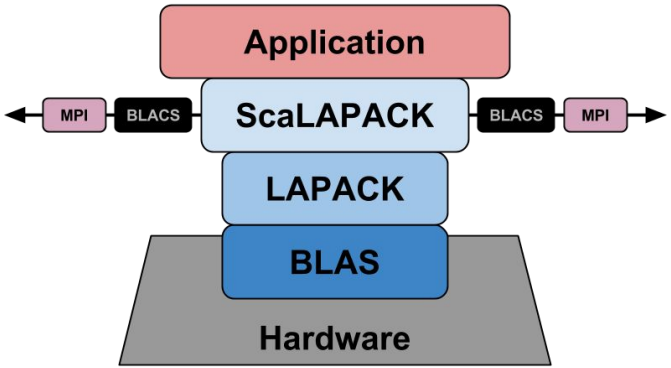


Charlene Yang
Leading



Our Messaging

1. **Understand your code performance. It's easier than you think and what you learn will help you everywhere.**
2. **Use Libraries/Frameworks when possible**



TensorFlow

Our Messaging

1. **Understand your code performance. It's easier than you think and what you learn will help you everywhere.**
2. **Use Libraries when possible**
3. **Abstractions: Identify and use appropriate abstractions to flexibly expose the parallelism in a problem.**

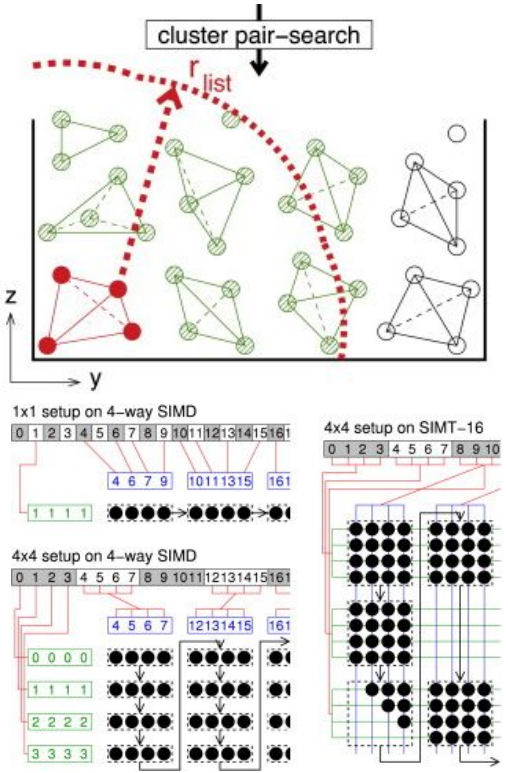
Abstractions and parallelism

Abstract operations for variable-width vectors

Example: Gromacs “cluster” pair-list adapts to 128, 256, 512-bit simd and 32 way SIMT by resizing the cluster.

Porting to new arch = implement abstract interface with intrinsics

***Effectiveness of this strategy depends on number of performance critical kernels**



Our Messaging

1. **Understand your code performance. It's easier than you think and what you learn will help you everywhere.**
2. **Use Libraries**
3. **Abstractions: Identify and use appropriate abstractions to flexibly expose the parallelism in a problem**
4. **Use a Programming Model that Helps You Abstract**

   Kokkos, Raja, OpenMP, OpenACC etc.

Summary

We are really Excited for Perlmutter's arrival later this year.

There is high potential for performance portability across Perlmutter, Aurora, Frontier, El Capitan!